Data-flow analysis

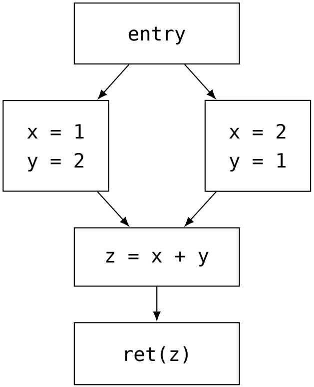
Data-flow analysis

Data-flow analysis

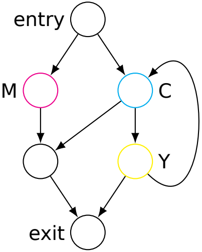
- Static
- Global (whole CFG)
- Control-flow dependent
- Computes run-time properties
- Unified formal model and theory

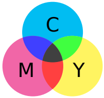
Applications

- Reaching definitions (use-def links)
- Live-variable analysis
- Constant propagation
- Constant subexpression elimination
- Dead code elimination



Example

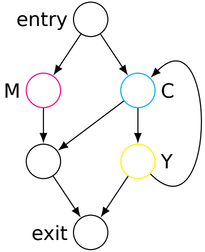


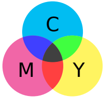


Example

Data-flow framework

- Flow graph $G = \langle V, E, v_{entry}, v_{exit} \rangle$
- Direction of analysis $D \in \{\downarrow, \uparrow\}$
- Meet-semilattice $\langle L, \wedge \rangle$ with upper bound
- Transfer functions $f_{v \in V} : L \to L$
- Boundary condition $in_0(v) = out_0(v) = \top$

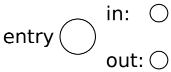


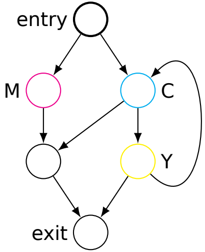


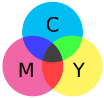
Example

Data-flow framework

- Flow graph $G = \langle V, E, v_{entry}, v_{exit} \rangle$
- Direction of analysis $D \in \{\downarrow, \uparrow\}$
- Meet-semilattice $\langle L, \wedge \rangle$ with upper bound
- Transfer functions $f_{v \in V} : L \to L$
- Boundary condition $in_0(v) = out_0(v) = \top$



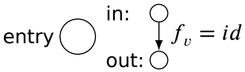


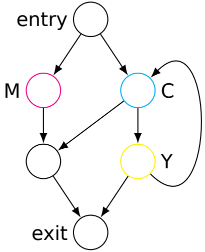


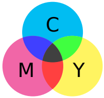
Example

Data-flow framework

- Flow graph $G = \langle V, E, v_{entry}, v_{exit} \rangle$
- Direction of analysis $D \in \{\downarrow, \uparrow\}$
- Meet-semilattice $\langle L, \wedge \rangle$ with upper bound
- Transfer functions $f_{v \in V} : L \to L$
- Boundary condition $in_0(v) = out_0(v) = \top$



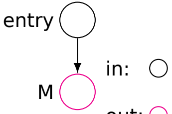


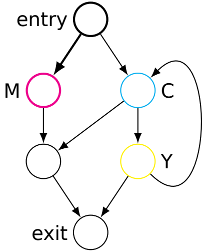


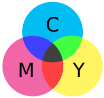
Example

Data-flow framework

- Flow graph $G = \langle V, E, v_{entry}, v_{exit} \rangle$
- Direction of analysis $D \in \{\downarrow, \uparrow\}$
- Meet-semilattice $\langle L, \wedge \rangle$ with upper bound
- Transfer functions $f_{v \in V} : L \to L$
- Boundary condition $in_0(v) = out_0(v) = \top$



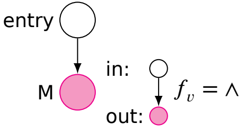


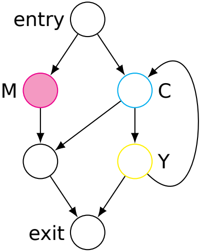


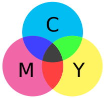
Example

Data-flow framework

- Flow graph $G = \langle V, E, v_{entry}, v_{exit} \rangle$
- Direction of analysis $D \in \{\downarrow, \uparrow\}$
- Meet-semilattice $\langle L, \wedge \rangle$ with upper bound
- Transfer functions $f_{v \in V} : L \to L$
- Boundary condition $in_0(v) = out_0(v) = \top$



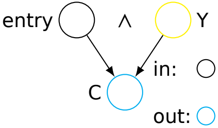


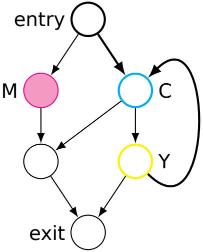


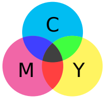
Example

Data-flow framework

- Flow graph $G = \langle V, E, v_{entry}, v_{exit} \rangle$
- Direction of analysis $D \in \{\downarrow, \uparrow\}$
- Meet-semilattice $\langle L, \wedge \rangle$ with upper bound
- Transfer functions $f_{v \in V} : L \to L$
- Boundary condition $in_0(v) = out_0(v) = \top$



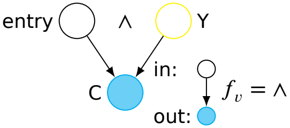


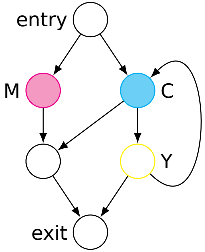


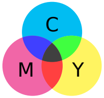
Example

Data-flow framework

- Flow graph $G = \langle V, E, v_{entry}, v_{exit} \rangle$
- Direction of analysis $D \in \{\downarrow, \uparrow\}$
- Meet-semilattice $\langle L, \wedge \rangle$ with upper bound
- Transfer functions $f_{v \in V} : L \to L$
- Boundary condition $in_0(v) = out_0(v) = \top$



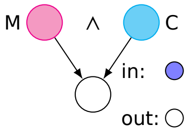


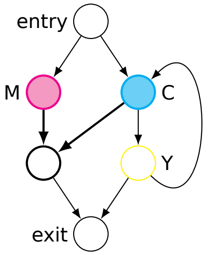


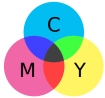
Example

Data-flow framework

- Flow graph $G = \langle V, E, v_{entry}, v_{exit} \rangle$
- Direction of analysis $D \in \{\downarrow, \uparrow\}$
- Meet-semilattice $\langle L, \wedge \rangle$ with upper bound
- Transfer functions $f_{v \in V} : L \to L$
- Boundary condition $in_0(v) = out_0(v) = \top$



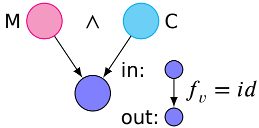


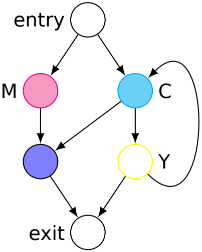


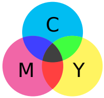
Example

Data-flow framework

- Flow graph $G = \langle V, E, v_{entry}, v_{exit} \rangle$
- Direction of analysis $D \in \{\downarrow, \uparrow\}$
- Meet-semilattice $\langle L, \wedge \rangle$ with upper bound
- Transfer functions $f_{v \in V} : L \to L$
- Boundary condition $in_0(v) = out_0(v) = \top$



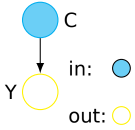


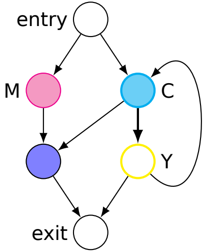


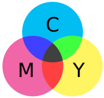
Example

Data-flow framework

- Flow graph $G = \langle V, E, v_{entry}, v_{exit} \rangle$
- Direction of analysis $D \in \{\downarrow, \uparrow\}$
- Meet-semilattice $\langle L, \wedge \rangle$ with upper bound
- Transfer functions $f_{v \in V} : L \to L$
- Boundary condition $in_0(v) = out_0(v) = \top$



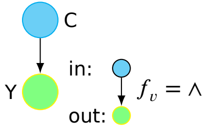


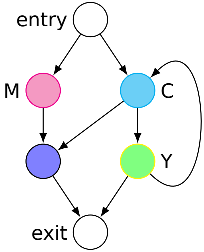


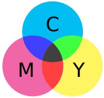
Example

Data-flow framework

- Flow graph $G = \langle V, E, v_{entry}, v_{exit} \rangle$
- Direction of analysis $D \in \{\downarrow, \uparrow\}$
- Meet-semilattice $\langle L, \wedge \rangle$ with upper bound
- Transfer functions $f_{v \in V} : L \to L$
- Boundary condition $in_0(v) = out_0(v) = \top$



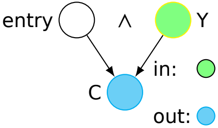


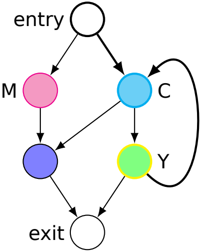


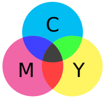
Example

Data-flow framework

- Flow graph $G = \langle V, E, v_{entry}, v_{exit} \rangle$
- Direction of analysis $D \in \{\downarrow, \uparrow\}$
- Meet-semilattice $\langle L, \wedge \rangle$ with upper bound
- Transfer functions $f_{v \in V} : L \to L$
- Boundary condition $in_0(v) = out_0(v) = \top$



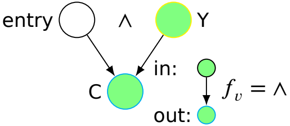


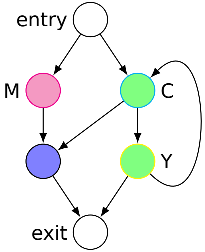


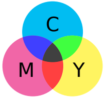
Example

Data-flow framework

- Flow graph $G = \langle V, E, v_{entry}, v_{exit} \rangle$
- Direction of analysis $D \in \{\downarrow, \uparrow\}$
- Meet-semilattice $\langle L, \wedge \rangle$ with upper bound
- Transfer functions $f_{v \in V} : L \to L$
- Boundary condition $in_0(v) = out_0(v) = \top$



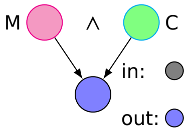


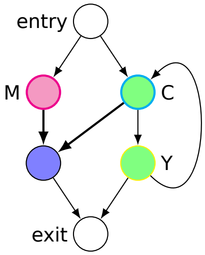


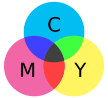
Example

Data-flow framework

- Flow graph $G = \langle V, E, v_{entry}, v_{exit} \rangle$
- Direction of analysis $D \in \{\downarrow, \uparrow\}$
- Meet-semilattice $\langle L, \wedge \rangle$ with upper bound
- Transfer functions $f_{v \in V} : L \to L$
- Boundary condition $in_0(v) = out_0(v) = \top$



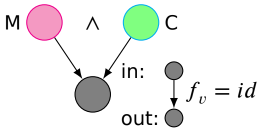


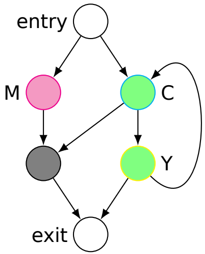


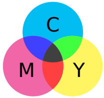
Example

Data-flow framework

- Flow graph $G = \langle V, E, v_{entry}, v_{exit} \rangle$
- Direction of analysis $D \in \{\downarrow, \uparrow\}$
- Meet-semilattice $\langle L, \wedge \rangle$ with upper bound
- Transfer functions $f_{v \in V} : L \rightarrow L$
- Boundary condition $in_0(v) = out_0(v) = \top$



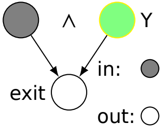


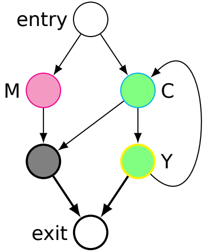


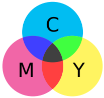
Example

Data-flow framework

- Flow graph $G = \langle V, E, v_{entry}, v_{exit} \rangle$
- Direction of analysis $D \in \{\downarrow, \uparrow\}$
- Meet-semilattice $\langle L, \wedge \rangle$ with upper bound
- Transfer functions $f_{v \in V} : L \to L$
- Boundary condition $in_0(v) = out_0(v) = \top$



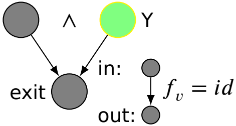


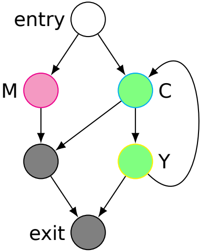


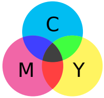
Example

Data-flow framework

- Flow graph $G = \langle V, E, v_{entry}, v_{exit} \rangle$
- Direction of analysis $D \in \{\downarrow, \uparrow\}$
- Meet-semilattice $\langle L, \wedge \rangle$ with upper bound
- Transfer functions $f_{v \in V} : L \rightarrow L$
- Boundary condition $in_0(v) = out_0(v) = \top$







Meet-semilattice

Binary operation $\wedge$ (*meet*)

- $x \wedge x = x$ (*idempotency*)
- $x \wedge y = y \wedge x$ (*commutativity*)
- $(x \wedge y) \wedge z = x \wedge (y \wedge z)$ (*associativity*)

Partial order $\leq$

- $x \leq x$ (*reflexivity*)
- $x \leq y \;\&\; y \leq z \Rightarrow x \leq z$ (*transitivity*)
- $x \leq y \;\&\; y \leq x \Rightarrow x = y$ (*antisymmetry*)

Semilattice $\langle L, \wedge \rangle$ [1] [2]

- $x \leq y \Leftrightarrow_{def} x \wedge y = x$
- $x < y \Leftrightarrow_{def} x \wedge y = x \;\&\; x \neq y$

[1] Do partial order conditions hold for this definition of $\leq$ via $\wedge$?

[2] Is it possible to define semilattice $\langle L, \wedge \rangle$ having only partial order $\langle L, \leq \rangle$?

Meet-semilattice

Binary operation $\wedge$ (*meet*)

- $x \wedge x = x$ (*idempotency*)
- $x \wedge y = y \wedge x$ (*commutativity*)
- $(x \wedge y) \wedge z = x \wedge (y \wedge z)$ (*associativity*)

Partial order $\leq$

- $x \leq x$ (*reflexivity*)
- $x \leq y \ \& \ y \leq z \Rightarrow x \leq z$ (*transitivity*)
- $x \leq y \ \& \ y \leq x \Rightarrow x = y$ (*antisymmetry*)

Semilattice $\langle L, \wedge \rangle$ [1] [2]

- $x \leq y \Leftrightarrow_{def} x \wedge y = x$
- $x < y \Leftrightarrow_{def} x \wedge y = x \ \& \ x \neq y$

Upper bound

$$\exists \bot \in L : \forall x \in L : \bot \wedge x = \bot \ (\bot \leq x)$$

Lower bound

$$\exists \top \in L : \forall x \in L : \top \wedge x = x \ (x \leq \top)$$

Semilattice height

$$H_L = max\left\{ |x_1 > x_2 > \ldots \in L| \right\}$$

Descending chain condition

$$\forall x_1 > x_2 > \ldots \in L : \exists k : \nexists y \in L : x_k > y$$

Semilattice product

$$\langle A, \wedge_A \rangle \times \langle B, \wedge_B \rangle = \langle A \times B, \wedge \rangle,$$
$$(a, b) \wedge (a', b') = (a \wedge_A a', b \wedge_B b')$$

---

[1] Do partial order conditions hold for this definition of $\leq$ via $\wedge$?

[2] Is it possible to define semilattice $\langle L, \wedge \rangle$ having only partial order $\langle L, \leq \rangle$?

Examples

Power set of S

$L = 2^S, \wedge = \cup$ or $\cap$

Examples

Power set of S

$L = 2^S, \wedge = \cup$ or $\cap$

$$\begin{tikzcd} \varnothing = \top \arrow[d] \\ \{A\} = \perp \end{tikzcd}$$

Examples

Power set of S

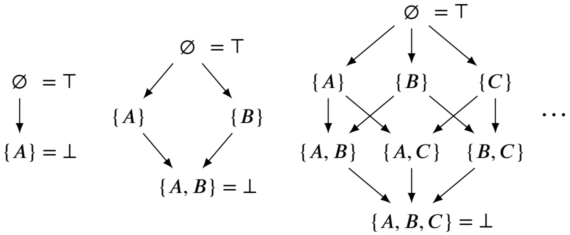
$L = 2^S, \wedge = \cup$ or $\cap$

$$\begin{tikzcd}
\varnothing = \top \arrow[d] \\
\{A\} = \bot
\end{tikzcd}
\qquad
\begin{tikzcd}
& \varnothing = \top \arrow[dl] \arrow[dr] & \\
\{A\} \arrow[dr] & & \{B\} \arrow[dl] \\
& \{A, B\} = \bot &
\end{tikzcd}$$

Examples

Power set of S

$L = 2^S$, $\wedge = \cup$ or $\cap$



Examples

Power set of S

$L = 2^S, \wedge = \cup$ or $\cap$

$$
\begin{tikzcd}
\varnothing = \top \arrow[d] \\
\{A\} = \bot
\end{tikzcd}
\qquad
\begin{tikzcd}
& \varnothing = \top \arrow[dl] \arrow[dr] & \\
\{A\} \arrow[dr] & & \{B\} \arrow[dl] \\
& \{A, B\} = \bot &
\end{tikzcd}
\qquad
\begin{tikzcd}
& \varnothing = \top \arrow[dl] \arrow[d] \arrow[dr] & \\
\{A\} \arrow[d] \arrow[dr] & \{B\} \arrow[dl] \arrow[dr] & \{C\} \arrow[dl] \arrow[d] \\
\{A, B\} \arrow[dr] & \{A, C\} \arrow[d] & \{B, C\} \arrow[dl] \\
& \{A, B, C\} = \bot &
\end{tikzcd}
\qquad \cdots
$$

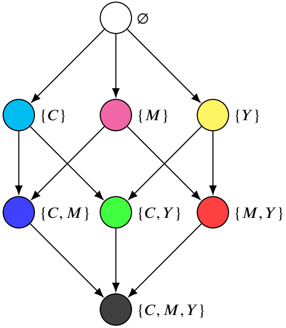
Examples

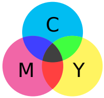
Power set of S

$L = 2^S, \wedge = \cup$ or $\cap$

CMYK

$L = 2^{\{C,M,Y\}}, \wedge = \cup$





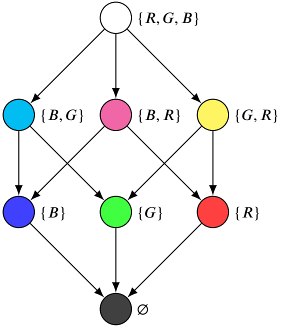
Examples

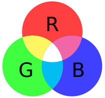
Power set of S

$L = 2^S, \wedge = \cup$ or $\cap$

RGB

$L = 2^{\{R,G,B\}}, \wedge = \cap$





Power set of S

$L = 2^S, \wedge = \cup$ or $\cap$

Natural numbers

$L = \mathbb{N}_0 \cup \{\top\}, x \wedge y = min(x, y)$

$$
\begin{tikzcd}
\top \arrow[d] \\
\cdots \arrow[d] \\
2 \arrow[d] \\
1 \arrow[d] \\
0
\end{tikzcd}
$$

Examples

Power set of S

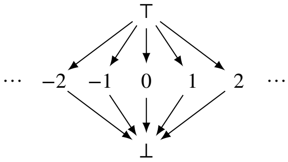
$L = 2^S, \wedge = \cup$ or $\cap$

Natural numbers

$L = \mathbb{N}_0 \cup \{\top\}, x \wedge y = min(x, y)$

Integer constants

$L = \mathbb{Z} \cup \{\top, \bot\}, \bot < \mathbb{Z} < \top$



Examples

Power set of S

$L = 2^S, \wedge = \cup$ or $\cap$

Natural numbers

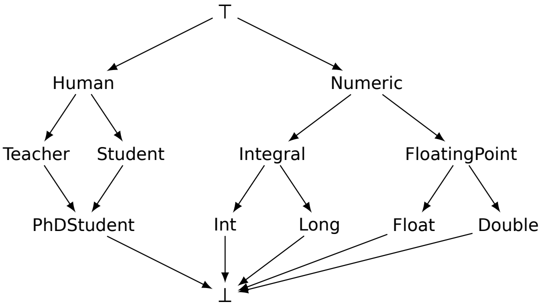
$L = \mathbb{N}_0 \cup \{\top\}, x \wedge y = min(x, y)$

Integer constants

$L = \mathbb{Z} \cup \{\top, \bot\}, \bot < \mathbb{Z} < \top$

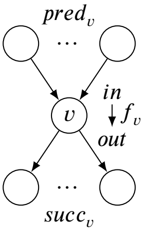
Program type hierarchy

$L = Types, x \leq y \Leftrightarrow x <: y$



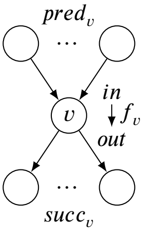
Data-flow framework

- Flow graph $G = \langle V, E, v_{entry}, v_{exit} \rangle$
- Direction of analysis $D \in \{\downarrow, \uparrow\}$
- Meet-semilattice $\langle L, \wedge \rangle$ with upper bound
- Transfer functions $f_{v \in V} : L \to L$
- Boundary condition $in_0(v) = out_0(v) = \top$



Data-flow framework

- Flow graph $G = \langle V, E, v_{entry}, v_{exit} \rangle$
- Direction of analysis $D \in \{\downarrow, \uparrow\}$
- Meet-semilattice $\langle L, \wedge \rangle$ with upper bound
- Transfer functions $f_{v \in V} : L \to L$
- Boundary condition $in_0(v) = out_0(v) = \top$



Data-flow equations

$D = \downarrow$	$D = \uparrow$
$in_0(v) = out_0(v) = \top$	$in_0(v) = out_0(v) = \top$
$in_i(v) = \bigwedge_{x \in pred_v} out_i(x)$	$out_i(v) = \bigwedge_{x \in succ_v} in_i(x)$
$out_i(v) = f_v(in_i(v))$	$in_i(v) = f_v(out_i(v))$

Maximum Fixed Point (MFP)

Maximum solution among all solutions S

$out_S(v) \leq out_{MFP}(v)$	$in_S(v) \leq in_{MFP}(v)$

Convergence conditions

- Monotonicity of transfer functions f_v
- Meet-semilattice $\langle L, \wedge \rangle$ with descending chains condition

Data-flow framework

Data-flow framework

- Flow graph $G = \langle V, E, v_{entry}, v_{exit} \rangle$
- Direction of analysis $D \in \{\downarrow, \uparrow\}$
- Meet-semilattice $\langle L, \wedge \rangle$ with upper bound
- Transfer functions $f_{v \in V} : L \to L$
- Boundary condition $in_0(v) = out_0(v) = \top$

Transfer functions

Monotone function f on $\langle L, \leq \rangle$

$$x \leq y \Rightarrow f(x) \leq f(y)$$

Monotone function f on $\langle L, \wedge \rangle$ [3]

$$f(x \wedge y) \leq f(x) \wedge f(y)$$

Distributive function f on $\langle L, \wedge \rangle$

$$f(x \wedge y) = f(x) \wedge f(y)$$

Data-flow equations

$D = \downarrow$	$D = \uparrow$
$in_0(v) = out_0(v) = \top$	$in_0(v) = out_0(v) = \top$
$in_i(v) = \bigwedge_{x \in pred_v} out_i(x)$	$out_i(v) = \bigwedge_{x \in succ_v} in_i(x)$
$out_i(v) = f_v(in_i(v))$	$in_i(v) = f_v(out_i(v))$

Maximum Fixed Point (MFP)

Maximum solution among all solutions S

$out_S(v) \leq out_{MFP}(v)$ | $in_S(v) \leq in_{MFP}(v)$

Convergence conditions

- Monotonicity of transfer functions f_v
- Meet-semilattice $\langle L, \wedge \rangle$ with descending chains condition

[3] Prove the equivalence of given monotone function definitions on $\langle L, \leq \rangle$ and on $\langle L, \wedge \rangle$.

Examples of divergent analysis

Monotonicity of transfer functions

$$L = \{T, F\}, F \leq T$$

$$f_{entry} = f_{exit} = id$$

$$f_{loop}(x) = \neg x$$

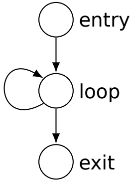
Descending chain condition [4]

$$L = \mathbb{R}_{\mathbb{0}}^{+} \cup \{\top\}, \wedge = min$$

$$f_{entry}(x) = 1$$

$$f_{loop}(x) = x/2$$

$$f_{exit} = id$$



[4] Can there be a semilattice with unlimited height which satisfies descending chain condition?

Meet Over Paths (MOP) [5]

Precise solution over all paths $v_{entry} \to \cdots \to v$

$$out_{MOP}(v) = \bigwedge_{v_{entry} \to \cdots \to v} f_v(\dots(f_{v_{entry}}(\top))\dots)$$

MFP safety

$$out_{MFP}(v) \leq out_{MOP}(v)$$

[5] Here we only consider forward-flow analysis $D = \downarrow$, the case of backwards-flow analysis $D = \uparrow$ is the same.

Meet Over Paths (MOP) [5]

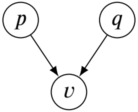
Precise solution over all paths $v_{entry} \to \cdots \to v$

$$out_{MOP}(v) = \bigwedge_{v_{entry} \to \cdots \to v} f_v(\ldots(f_{v_{entry}}(\top))\ldots)$$

MFP safety

$$out_{MFP}(v) \leq out_{MOP}(v)$$

$out_{MOP}(p) = x$ $\quad$ $out_{MOP}(q) = y$



[5] Here we only consider forward-flow analysis $D = \downarrow$, the case of backwards-flow analysis $D = \uparrow$ is the same.

Meet Over Paths (MOP) [5]

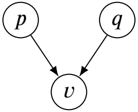
Precise solution over all paths $v_{entry} \to \cdots \to v$

$$out_{MOP}(v) = \bigwedge_{v_{entry} \to \cdots \to v} f_v(\ldots(f_{v_{entry}}(\top))\ldots)$$

MFP safety

$$out_{MFP}(v) \le out_{MOP}(v)$$

$out_{MOP}(p) = x$ $\quad$ $out_{MOP}(q) = y$



$$\underbrace{f_v(x \wedge y)}_{out_{MFP}(v)}$$

[5] Here we only consider forward-flow analysis $D = \downarrow$, the case of backwards-flow analysis $D = \uparrow$ is the same.

Precision

Meet Over Paths (MOP) [5]

Precise solution over all paths $v_{entry} \to \cdots \to v$

$$out_{MOP}(v) = \bigwedge_{v_{entry} \to \cdots \to v} f_v(\ldots(f_{v_{entry}}(\top))\ldots)$$

MFP safety

$$out_{MFP}(v) \leq out_{MOP}(v)$$

$$\begin{tikzcd} out_{MOP}(p) = x \quad p \arrow[dr] & & q \arrow[dl] \quad out_{MOP}(q) = y \\ & v & \end{tikzcd}$$

$$\underbrace{f_v(x \wedge y)}_{out_{MFP}(v)} \qquad \underbrace{f_v(x) \wedge f_v(x)}_{out_{MOP}(v)}$$

[5] Here we only consider forward-flow analysis $D = \downarrow$, the case of backwards-flow analysis $D = \uparrow$ is the same.

Precision

Meet Over Paths (MOP) [5]

Precise solution over all paths $v_{entry} \rightarrow \cdots \rightarrow v$

$$out_{MOP}(v) = \bigwedge_{v_{entry} \rightarrow \cdots \rightarrow v} f_v(\dots(f_{v_{entry}}(\top))\dots)$$

MFP safety

$$out_{MFP}(v) \leq out_{MOP}(v)$$

$$\begin{tikzcd} out_{MOP}(p) = x \quad p \arrow[dr] & & q \arrow[dl] \quad out_{MOP}(q) = y \\ & v & \end{tikzcd}$$

$$\underbrace{f_v(x \wedge y)}_{out_{MFP}(v)} \leq \underbrace{f_v(x) \wedge f_v(x)}_{out_{MOP}(v)}$$

[5] Here we only consider forward-flow analysis $D =\downarrow$, the case of backwards-flow analysis $D =\uparrow$ is the same.

Meet Over Paths (MOP) [5]

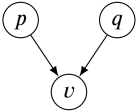
Precise solution over all paths $v_{entry} \to \cdots \to v$

$$out_{MOP}(v) = \bigwedge_{v_{entry} \to \cdots \to v} f_v(\dots(f_{v_{entry}}(\top))\dots)$$

MFP safety [6]

$$out_{MFP}(v) \leq out_{MOP}(v)$$

$out_{MOP}(p) = x$ $\quad$ $out_{MOP}(q) = y$



$$\underbrace{f_v(x \wedge y)}_{out_{MFP}(v)} \leq \underbrace{f_v(x) \wedge f_v(x)}_{out_{MOP}(v)}$$

[5] Here we only consider forward-flow analysis $D = \downarrow$, the case of backwards-flow analysis $D = \uparrow$ is the same.

[6] If transfer functions are distributive, then MFP is always precise — $out_{MFP}(v) = out_{MOP}(v)$.

Meet Over Paths (MOP) [5]

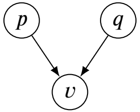
Precise solution over all paths $v_{entry} \to \cdots \to v$

$$out_{MOP}(v) = \bigwedge_{v_{entry} \to \cdots \to v} f_v(\dots(f_{v_{entry}}(\top))\dots)$$

MFP safety [6]

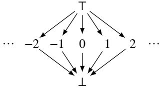
$$out_{MFP}(v) \le out_{MOP}(v)$$

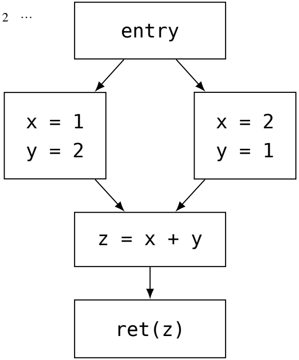
$out_{MOP}(p) = x$ $\qquad$ $out_{MOP}(q) = y$



$$\underbrace{f_v(x \wedge y)}_{out_{MFP}(v)} \le \underbrace{f_v(x) \wedge f_v(x)}_{out_{MOP}(v)}$$

Constant propagation





[5] Here we only consider forward-flow analysis $D = \downarrow$, the case of backwards-flow analysis $D = \uparrow$ is the same.

[6] If transfer functions are distributive, then MFP is always precise — $out_{MFP}(v) = out_{MOP}(v)$.

Meet Over Paths (MOP) [5]

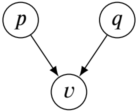
Precise solution over all paths $v_{entry} \to \cdots \to v$

$$out_{MOP}(v) = \bigwedge_{v_{entry} \to \cdots \to v} f_v(\ldots(f_{v_{entry}}(\top))\ldots)$$

MFP safety [6]

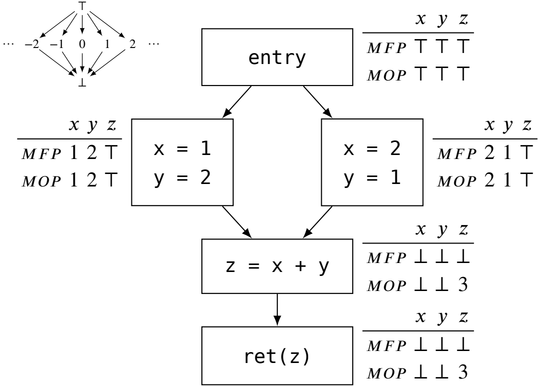
$$out_{MFP}(v) \leq out_{MOP}(v)$$

$out_{MOP}(p) = x$ $\quad$ $out_{MOP}(q) = y$



$$\underbrace{f_v(x \wedge y)}_{out_{MFP}(v)} \leq \underbrace{f_v(x) \wedge f_v(x)}_{out_{MOP}(v)}$$

Constant propagation



[5] Here we only consider forward-flow analysis $D = \downarrow$, the case of backwards-flow analysis $D = \uparrow$ is the same.

[6] If transfer functions are distributive, then MFP is always precise — $out_{MFP}(v) = out_{MOP}(v)$.

Control-flow graph

- $CFG = \langle B, E, entry, exit \rangle$
- Every block $b \in B$ contains exactly one operation
- V — set of varbles in a program
- $def_v \subseteq B$ — set of *assignments* into variable $v \in V$ (e.g. `v = 3`)
- $use_v \subseteq B$ — set of *uses* of variable $v \in V$ (e.g. `x = y + v`)

Data-flow analysis of programs

Control-flow graph

- $CFG = \langle B, E, entry, exit \rangle$
- Every block $b \in B$ contains exactly one operation
- V — set of varbles in a program
- $def_v \subseteq B$ — set of *assignments* into variable $v \in V$ (e.g. `v = 3`)
- $use_v \subseteq B$ — set of *uses* of variable $v \in V$ (e.g. `x = y + v`)

Gen-Kill formalism

- $L = 2^S$, $\wedge = \cup$ or $\cap$
- $f_b(x) = gen_b \cup (x \setminus kill_b)$
- gen_b — properties *generated* by block b
- $kill_b$ — properties *killed* by block b

Data-flow analysis of programs

Control-flow graph

- $CFG = \langle B, E, entry, exit \rangle$
- Every block $b \in B$ contains exactly one operation
- V — set of varbles in a program
- $def_v \subseteq B$ — set of *assignments* into variable $v \in V$ (e.g. `v = 3`)
- $use_v \subseteq B$ — set of *uses* of variable $v \in V$ (e.g. `x = y + v`)

Gen-Kill formalism

- $L = 2^S$, $\wedge = \cup$ or $\cap$
- $f_b(x) = gen_b \cup (x \setminus kill_b)$
- gen_b — properties *generated* by block b
- $kill_b$ — properties *killed* by block b

Result

- $\langle L, \wedge \rangle$ — finite semilattice
- f_b — distributive functions [7]
- Analysis *always* converges to precise solution

[7] Prove distributivity of f_b in gen-kill form.

Control-flow graph

- $CFG = \langle B, E, entry, exit \rangle$
- Every block $b \in B$ contains exactly one operation
- V — set of varbles in a program
- $def_v \subseteq B$ — set of *assignments* into variable $v \in V$ (e.g. `v = 3`)
- $use_v \subseteq B$ — set of *uses* of variable $v \in V$ (e.g. `x = y + v`)

Reaching definitions

$$L = 2^B, \wedge = \cup, D = \downarrow$$

b	$\in def_v$	$\notin def_v$
gen_b	$\{b\}$	$\varnothing$
$kill_b$	def_v	$\varnothing$

Live-variable analysis

$$L = 2^V, \wedge = \cup, D = \uparrow$$

gen_b	$\{v \mid b \in use_v\}$
$kill_b$	$\{v \mid b \in def_v\}$

Gen-Kill formalism

- $L = 2^S, \wedge = \cup$ or $\cap$
- $f_b(x) = gen_b \cup (x \setminus kill_b)$
- gen_b — properties *generated* by block b
- $kill_b$ — properties *killed* by block b

Result

- $\langle L, \wedge \rangle$ — finite semilattice
- f_b — distributive functions [7]
- Analysis *always* converges to precise solution

[7] Prove distributivity of f_b in gen-kill form.

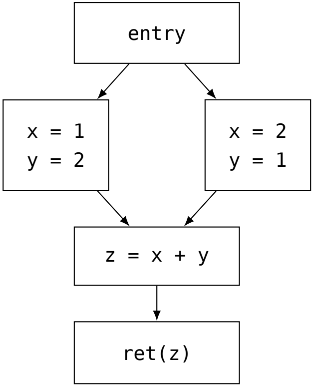
Conclusion

Benefits

- Global static analysis
- Universal theoretical model
- Straightforward implementation
- gen-kill formalism guarantees convergence and precision

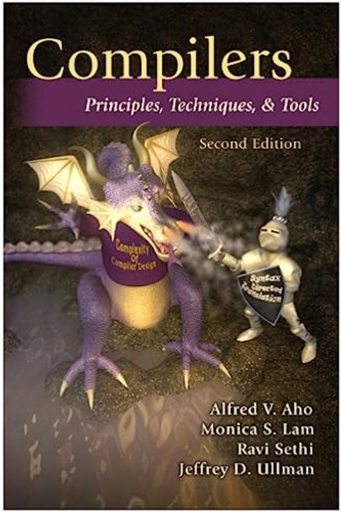
Drawbacks

- Result gets invalidated after optimizations
- Analsyses do not compose quite efficiently
- Convergence and precision are not guaranteed in general



Further reading

A. V. Aho, M. S. Lam, R. Sethi, and J. D. Ullman.
Compilers: Principles, Techniques, and Tools, 1986
Introduction to Data-Flow Analysis

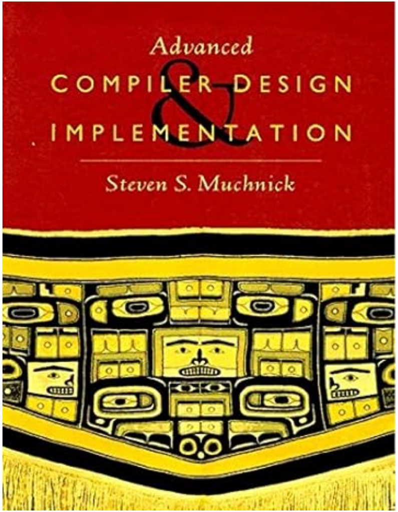


Further reading

S. S. Muchnick.

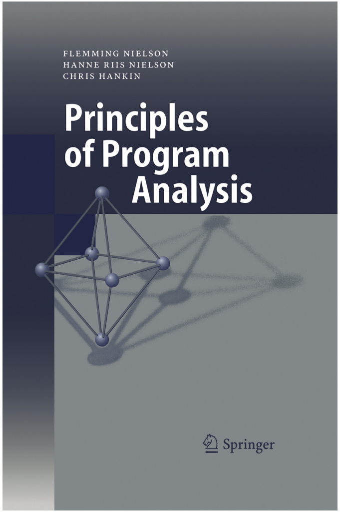
Advanced compiler design and implementation, 1997

Data-Flow Analysis



Further reading

N. Flemming, H. R. Nielson, and C. Hankin.
Principles of program analysis, 2015
Data Flow Analysis



Thank you for attention